To appear in the Proceedings of 1997 ICRA

A Motion Planning Based Approach for Inverse Kinematics of Redundant Robots: The Kinematic Roadmap

Juan Manuel Ahuactzin*
jmal@udlapvms.pue.udlap.mx
Department of Computer Science
Universidad de las Américas-Puebla
72820 Cholula, Puebla Mexico

Kamal Gupta
kamal@cs.sfu.ca
School of Engineering Science
Simon Fraser University
Burnaby, B.C. Canada V5A 1S6

Abstract

We propose a new approach to solving the point-to-point inverse kinematics problem for highly redundant manipulators. It is inspired by recent motion planning research and explicitly takes into account constraints due to joint limits and self-collisions. Central to our approach is the novel notion of *kinematic roadmap* for a manipulator. The kinematic roadmap captures the connectivity of the configuration space of a manipulator in a finite graph like structure. The standard formulation of inverse kinematics problem is then solved using this roadmap. Our current implementation, based on Ariadne's Clew Algorithm [BA+93], is composed of two sub-algorithms: *EXPLORE*, an appealingly simple algorithm that builds the kinematic roadmap by placing landmarks in the configuration space; and *SEARCH*, a local planner, that uses this roadmap to reach the desired end-effector configuration. Our implementation of *SEARCH* is an extremely efficient closed form solution, albeit local, to inverse kinematics that exploits the serial kinematic structure of serial manipulator arms. Initial experiments with a 7-*dof* manipulator have been extremely successful.

1 Introduction

Kinematically redundant robots ($>$ 6-*dof*, since a general positioning task may need up to 6 degrees of freedom) are finding increasing use in the robotics research community. Their main appeal is that the additional degrees of freedom can be used to avoid singularities and collisions, and to optimize performance criteria. However, with the increased degrees of freedom comes increased computational complexity for inverse kinematics and for motion planning and obstacle avoidance. Former, because closed-form inverse kinematics solutions don't exist in general, and the latter because of the increase in the dimensionality of the configuration space.

The inverse kinematic problem has been formulated in primarily two ways. The first one – we call it *point-to-point kinematic problem* – is to compute a robot configuration corresponding to a given end-effector configuration. Formally, it is stated as solving the equation :

$$f^{-1}(\hat{x}_g) = \hat{q} \tag{1}$$

where $\hat{q} \in I\!R^n$ denotes a robot configuration (joint vector) and $\hat{x}_g \in I\!R^m$ denote the desired configuration of the end-effector in work space ad $f : I\!R^n \rightarrow I\!R^m$ is the forward kinematic function.

The second one, also called *kinematic control problem* [Sic90], is to find an entire joint space trajectory $\hat{q}(t)$ corresponding to a given work space trajectory, $\hat{x}_g(t),\ t \in [0, T]$ of the end-effector such that

$$f^{-1}(\hat{x}_g(t)) = \hat{q}(t) \quad t \in [0, T] \tag{2}$$

There are two main aspects of solution to inverse kinematic problem: motion planning, which deals with *existence* of a path, and redundancy resolution, which deals with selecting a single configuration among a set of many possible ones (this set is the *self-motion manifold*) [LL94]. Most research in inverse kinematics algorithms has primarily focussed on local algorithms for redundancy resolution [Sic90]. These approaches address the problem in the velocity domain, i.e., they use the linearized mapping $\dot{x} = J(\hat{q})\dot{\hat{q}}$, where J is the $(n \times m)$ Jacobian matrix. These approaches treat hard kinematic constraints by transforming them into some sort of potential functions and can be easily caught in local minima. In motion planning terminology, there is no guarantee of *completeness*. Furthermore, most such approaches completely ignore the constraints due to *self-collisions* among various links of the manipulators. Note that research in motion planning[1], on the other hand, has dealt with such constraints in a very different and effective manner by explicitly characterizing the free regions in the configuration space (joint space) [Lat91].

Tools from topology have been employed to characterize the global set of solutions for point-to-point inverse kinematics – in particular, *c-bundles* and *self-motion manifolds* – [Bur89]. *These approaches are mainly aimed at characterizing the global set of solutions rather than finding a solution if there exists one.* Based on [Bur89], [LL94] discusses a framework for global path planning, that attempts to systematically characterize different levels of planning for inverse kinematics. At the top level it assumes that a graph like compact description of the configuration space is available, and it is suggested that an A^* type algorithm could be used for searching this graph. However, the topology of the configuration space (with joint limits, self-collisions and possibly obstacle constraints) can be extremely compli-

*The author was a visiting research scientist at Simon Fraser University during Summer 1996. Research was funded by NSERC research and CRD grants.

[1] In motion planning problem, the initial and final configurations of the robot are specified and the aim is to find a collision-free path connecting the two [Lat91].

cated and it is not clear how to obtain this graph like structure in general. In the motion planning literature, several much more effective approaches have been suggested to search high dimensional spaces with extremely complicated topological structure [Lat91, CH92, Gup90, GZ95, KL94, BA+93]. In particular, an *appealingly simple* emerging paradigm (although there are differences in specific details) in solving classical motion planning problems is to capture the connectivity of the configuration space using a finite (but possibly large) set of nodes (or landmarks) – generally called a *roadmap* [Lat91, BA+93, KL94].

In this paper, we propose that this *roadmap* paradigm is eminently suitable for inverse kinematics problem and apply this paradigm to solve the point-to-point inverse kinematics problem for redundant manipulators in the presence of joint limits, self-collisions and obstacle constraints. It makes absolute sense to capture the connectivity of the configuration space – in the presence of joint limits and self-collisions – in what we call the *kinematic roadmap* of the manipulator, since the kinematic structure of a manipulator does not change and once built, this can be used to quickly find *a solution* for inverse kinematics problem. Note that the emphasis of our approach is on the motion planning aspects of the problem, i.e., finding a path if there exists one. Once a feasible path is found, more elaborate optimization criteria can then be used to optimize it.

1.1 Overview of Our Approach

We first formulate the inverse kinematic problem as an optimization problem (as is normally done in the kinematics literature) over the configuration space of the robot, however, with the objective function being a somewhat novel metric between the initial and the desired end effector frames. This optimization problem is then solved using a framework similar to *Ariadne's Clew Algorithm* (*ACA*), originally proposed for classical motion planning [BA+93, Ahu94]. Our approach is best described as composed of two sub-algorithms: *EXPLORE* and *SEARCH*, executed in an interleaved manner. The *EXPLORE* algorithm "explores" the reachable configuration space from the given initial point by iteratively placing "landmarks" in it. The landmarks are placed so that a path from the initial position to any landmark is known. The set of landmarks with the associated paths is a *kinematic roadmap* that represents the connectivity of the configuration space in the presence of kinematic constraints. The *SEARCH* algorithm is a local planner that verifies if the goal configuration can be reached from a newly placed landmark. *SEARCH* exploits the serial kinematic structure of a manipulator, and solves the optimization problem sequentially, in a closed form, thereby making it extremely efficient. Figure 1 shows a schematic representation of the process. Each • denotes a landmark L_i, with L_0 being the start configuration. The continuous paths show the trajectories used by *EXPLORE* to place a landmark and the dotted ones show the attempts by *SEARCH* to reach the goal position. The tip of the arrows shows how far the local planner *SEARCH* could proceed. In the schematic, *SEARCH* succeeds from L_{11}. The final trajectory will be then constructed using landmarks (and the corresponding paths connecting them) L_0,L_2,L_9,L_{11} and the path to the goal point found by *SEARCH*. The resolution completeness of the *ACA* approach for classical motion planning has been shown in [Ahu94]. We conjecture that a similar claim can be made for the inverse kinematics problem. However, our emphasis, in this paper, is on demonstrating the practical effectiveness of our approach.

We have implemented this approach for a 7-*dof* manipulator arm (with polyhedral models for links and joint-limits) shown in Figure 6, and our initial experiments have been extremely successful (See Section 6 for details). The planner always found a feasible path. A main reason for the success of our approach appears to be that *SEARCH* has a rather large pre-image region – the set of all points $\hat{q}$ such that *SEARCH* succeeds in getting to the goal, $\hat{x}_g$. Therefore the probability that *EXPLORE* will place a landmark in this pre-image region is quite high.

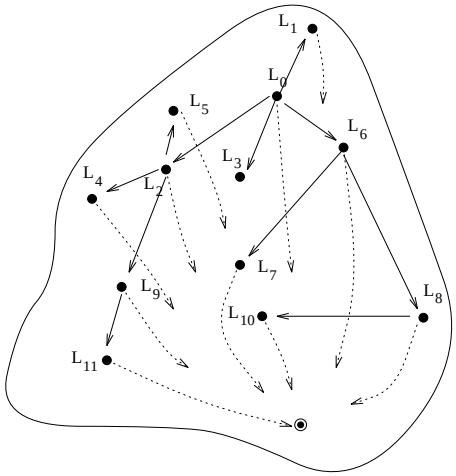


Figure 1: Schematic representation of the kinematic roadmap. The set of landmarks and the associated paths capture the connectivity of the configuration space in the presence of *self-collisions* and joint limits.

In summary, the salient features of our algorithm are: (i) it proposes and builds a novel representation – the kinematic roadmap – of the configuration space of the manipulator in a finite number of nodes; (ii) hard kinematic constraints due to joint limits, self-collisions (and other static obstacles) are explicitly taken into account while building this kinematic roadmap. The roadmap is then used to efficiently find *a feasible solution* if there exists one; and (iii) it avoids computation of Jacobian, often an expensive computation.

2 The Metric

2.1 Notation

Let $\mathcal{A}$ denote the robot arm, $\hat{q} = (q_1, q_2 \ldots q_n)$ denote a robot configuration, $\mathcal{G}$ denote the end-effector of $\mathcal{A}$, and $\hat{x} = (x, y, z, \psi, \theta, \gamma)$ denote a configuration of the end-effector, where ψ, θ, γ denote an appropriate parameterization of $SO(3)$. $\mathcal{C}_{\mathcal{A}}$ and $\mathcal{C}_{\mathcal{G}}$ denote the configuration spaces of $\mathcal{A}$ and $\mathcal{G}$ respectively. Following standard D-H notation in [Cra91], we use $\mathcal{F}_i$ to denote a frame attached to i^{th} joint, and ${}^aT_b$ is the homogenous transformation matrix (or simply matrix) that represents $\mathcal{F}_b$ w.r.t. $\mathcal{F}_a$. The trailing superscript may be omitted if the base frame is implicitly understood.

$T(\hat{x})$ represents the matrix corresponding to a configuration $\hat{x}$ of the end-effector frame $\mathcal{F}_G$.

2.2 The Metric

One could use any number of metrics between two configurations of the end-effector as the objective function, however, it is well known that these metrics mix translational and rotational components [Lat91]. Although there is no natural way of doing it in general since any distance metric in $SE(3)$ will ultimately depend on a choice of length scale [Par95], what we need is a way of measuring distance between two configurations of the same object and such metrics have been proposed before [KR92]. These metrics, however, are computationally expensive [Par95]. Instead, we define a novel metric d[2] between two coordinate frames, say $\mathcal{F}_a$ and $\mathcal{F}_b$ represented (w.r.t. some common frame $\mathcal{F}_c$) by matrices ${}^cT_a$ and ${}^cT_b$, as the sum of distances between the corresponding unit vectors along the co-ordinate axes, i.e.,

$$d({}^cT_a, {}^cT_b) = d_x + d_y + d_z \tag{3}$$

where d_x, d_y and d_z are the distances between the unit vectors along the x,y and z axes, respectively (see Figure 2). Note that trailing superscript may be dropped since this metric d is left-invariant, i.e., it is independent of the base frame $\mathcal{F}_c$.

This metric is then used to pose the inverse kinematics problem as an optimization problem. A cost function c is first defined as:

$$c(\hat{q}, \hat{x}_g) = d(T(f(\hat{q})), T(\hat{x}_g)) \tag{4}$$

and the inverse kinematic problem can then be posed as follows:

$$f^{-1}(\hat{x}_g) = \hat{q}_\bullet : \min_{\hat{q} \in \mathcal{C}_\mathcal{A}} c(\hat{q}, \hat{x}_g) \tag{5}$$

Note that $\hat{q} \in f^{-1}(\hat{x}_g) \Leftrightarrow c(\hat{q}, \hat{x}_g) = 0$. In the next two sections, we show an algorithm to solve this problem within the *ACA* paradigm. First we present a local algorithm (*SEARCH*) that computes a local minimum of the optimization problem, and then we present *EXPLORE* that spreads landmarks over the configuration space.

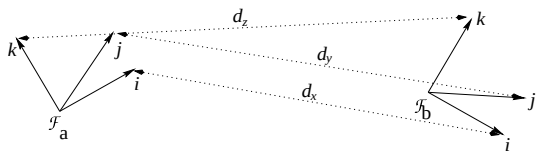


Figure 2: d_x, d_y and d_z shown graphically. Sum of these three quantities defines the metric d between two frame, $\mathcal{F}_a$ and $\mathcal{F}_b$.

3 The Local Algorithm *SEARCH*

Our approach exploits the serial kinematic structure of manipulator arms [Gup90, GZ95] to construct an efficient local algorithm to solve the optimization problem posed in the previous section. Let $\hat{q} = (q_1, q_2, ..., q_i, ..., q_n)$ be a free configuration of the robot. We denote by $\Delta_i = [\Delta_i^{min}, \Delta_i^{max}] \subset R$ the collision-free interval of joint i at $\hat{q}$ (see Figure 3). The joint limits are naturally represented as an interval and the collision constraints are easily computed as an interval by simple computational geometric methods [LP87]. The intersection of these intervals then gives the desired interval Δ_i which determines the feasible range of motion for joint i.

Formally, let $\mathcal{C}^i_{\hat{q}} \subset \mathcal{C}_{\mathcal{A}free}$ denote this feasible interval set for joint i, i.e.,

$$\mathcal{C}^i_{\hat{q}} = \{(q_1, q_2, ..., q_{i-1}, q_i + \delta, q_{i+1}, ..., q_n) | \delta \in \Delta_i\}$$

Let $\hat{x}_g \in \mathcal{C}_\mathcal{G}$ be the desired configuration of $\mathcal{G}$. For an arbitrary $\hat{q} \in \mathcal{C}_{\mathcal{A}free}$, a function g is defined as follows :

$$g(\hat{q}, i) = \hat{q}^{min} : \min_{\hat{q} \in \mathcal{C}^i_{\hat{q}}} c(\hat{q}, \hat{x}_g) \tag{6}$$

The configuration in $\mathcal{C}^i_{\hat{q}}$ that minimizes the cost function c is then $\hat{q}^{min}$. For brevity, we have not explicitly shown $\hat{x}_g$ as one of the arguments of $g(\cdot)$. Note that the values of joints other than i do not change as a result of applying $g(\hat{q}, i)$, i.e., if $\hat{q} = (q_1, q_2, ..., q_i, ..., q_{n-1}, q_n)$, then $g(\hat{q}, i) = (q_1, q_2, ..., q_i^{min}, ..., q_{n-1}, q_n)$.

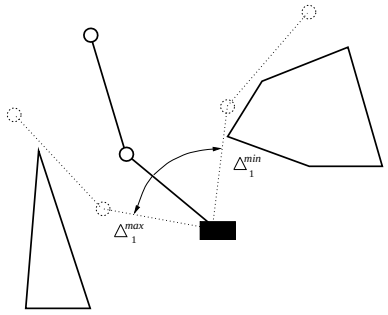


Figure 3: Illustration of collision-free interval $[\Delta_1^{min}, \Delta_1^{max}]$ due to a physical obstacle. Constraints due to joint limits and self-collisions are also represented as an interval.

From a given initial configuration, $\hat{q}_\circ = (q_1^0, q_2^0, ..., q_n^0) \in \mathcal{C}_\mathcal{A}$, $g(\hat{q}, i)$ is repeatedly applied in an iterative manner, i.e., apply $g(\hat{q}_0, 1)$ obtaining $\hat{q}_1 = (q_1^{min}, q_2^0, ..., q_n^0) \in \mathcal{C}^1_{\hat{q}_0}$, then apply $g(\hat{q}_1, 2)$ obtaining $\hat{q}_2 = (q_1^{min}, q_2^{min}, q_2^0..., q_n^0) \in \mathcal{C}^2_{\hat{q}_1}$ and so on until $\hat{q}_n = (q_1^{min}, q_2^{min}, q_2^{min}..., q_n^{min}) \in \mathcal{C}^n_{\hat{q}_{n-1}}$ is obtained. We adopt the notation :

$$g^n(\hat{q}_\circ) = g(g(g(...(g(\hat{q}_\circ, 1), 2)..., n-2), n-1), n) \tag{7}$$

Stated algorithmically, we can now define the *SEARCH* algorithm as follows:

SEARCH$(\hat{q}_0, \hat{x}_g)$
begin
  $\hat{q} = g^n(\hat{q}_0)$
  while $(\hat{q} \neq \hat{q}_0)$ *SEARCH*$(\hat{q}, \hat{x}_g)$
  return $c(\hat{q}, \hat{x}_g)$
end

Starting from $\hat{q} = \hat{q}_0$, *SEARCH* repeatedly applies $g^n(\hat{q})$, each time using the result of previous iteration as the starting point for the current iteration. Note that $\hat{q}_{loc} = g^{loc}(\hat{q}_\circ)$, is a global minimum (i.e. a solution to the inverse kinematic problem) if

[2] It is relatively straightforward to show that d is indeed a metric.

$SEARCH(\hat{q}, \hat{x}_g) = 0$, else a local minimum has been found.

We now describe how $g(\hat{q}, i)$ is implemented (for a revolute joint, the case for prismatic joint is similar). It is analytically derived by symbolically differentiating the cost function $c(\cdot)$ in equation 4 *w.r.t.* q_i. It is computationally advantageous to represent the goal frame w.r.t. frame $\mathcal{F}_i$. Let $P_i = (x_i, y_i, z_i), i = 1 \ldots 3$ denote the vectors that represent the tips of unit axes vectors ($\hat{i}, \hat{j}$ and $\hat{k}$) of $\mathcal{F}_\mathcal{G}$ w.r.t. $\mathcal{F}_i$ in configuration $\hat{x}_g$, i.e. $T(\hat{x}_g)$. Similarly, $P_i' = (x_i', y_i', z_i'), i = 1 \ldots 3$ denote the vectors that represent the tips of unit axes vectors $\mathcal{F}_\mathcal{G}$ w.r.t. $\mathcal{F}_i$ (see Figure 4) in an arbitrary configuration $\hat{q}$ of the robot. With this formualtion, the z coordinate (for a revolute joint, hence we explicitly use θ_i instead of q_i) remains constant and therefore simplifies the symbolic differentiation of the metric d. For example, in the expression for d_x in 3,

$$\begin{aligned} d_x^2 = \quad & (x_1 - x_1' * cos(\theta_i) - y_1' * sin(\theta_i))^2 + \\ & (y_1 - x_1' * sin(\theta_i) + y_1' * cos(\theta_i))^2 + \\ & (z_1 - z_1')^2 \end{aligned}$$

the term $(z_1 - z_1')^2$ is a constant and therefore has no influence in the optimization.

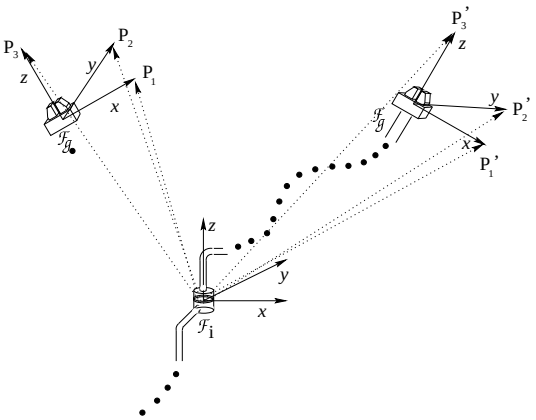


Figure 4: Vectors P_i and P_i' corresponding to the desired goal configuration and an arbitrary configuration of the end-effector. Both are defined w.r.t. $\mathcal{F}_i$.

Differentiating $d({}^iT(\hat{x}_g), {}^iT(f(\hat{q}))$ w.r.t. q_i and then equating the result to zero (using MAPLE) yields the following expression for the optimum value ($q_i = \theta_i^\star$) for a revolute joint:

$$\arctan\left(\frac{-x_1\, y_1' + y_1\, x_1' - y_2'\, x_2 + x_2'\, y_2 - y_3'\, x_3 + x_3'\, y_3}{x_1\, x_1' + y_1\, y_1' + x_2'\, x_2 + y_2'\, y_2 + x_3'\, x_3 + y_3'\, y_3}\right)$$

The global minimum of $c(\cdot)$ over the interval Δ_i is either at $\theta_i^\star$, or $\theta_i^\star + \pi$, or (if $\theta_i^\star$ lies outside the valid range Δ_i) one of the endpoints of the interval. This is the value returned by $g(\hat{q}, i)$.

4 Algorithm *EXPLORE*

The goal of $EXPLORE$ is to capture the connectivity of the accessible space from an initial configuration $\hat{q}_\circ \in \mathcal{C}_{\mathcal{A}free}$. A representation of that space is obtained by generating free paths and placing landmarks all over the configuration space beginning from an initial landmark (configuration) $L_1 = \hat{q}_\circ$. The structure obtained by $EXPLORE$ is essentially a tree where the set of nodes represents accessible configurations of $\mathcal{C}_{\mathcal{A}free}$ and the set of links represents free paths connecting these configurations. This structure is a *kinematic roadmap*.

$EXPLORE$ may be implemented in several ways. Here we present an appealingly simple but effective implementation that has been very successful in our experiments. It executes random paths, one each from the current set of landmarks. Each of these paths lie completely in $\mathcal{C}_{\mathcal{A}free}$. The end points of these random paths are called *embryos*. The embryo that is the *farthest* from the current set of landmarks is then chosen as the new landmark, and the process is repeated.

Formally, at the m^{th} iteration, let $\mathcal{L}_m$ denote the set of existing landmarks, and let $\mathcal{PL}_m = \{\hat{\varphi}_2, \hat{\varphi}_3, ..., \hat{\varphi}_j, ..., \hat{\varphi}_m\}$ be the set of paths from one landmark to another. Let $\mathcal{E}_m = \{\hat{e}_m, \hat{e}_{m-1}, ..., \hat{e}_1\}$ denote the set of existing embryos, and $\mathcal{PE}_m = \{\hat{\gamma}_1, \hat{\gamma}_2, ..., \hat{\gamma}_j, ..., \hat{\gamma}_m\}$ denote the set of paths going from landmarks to their corresponding embryos, where $\hat{\gamma}_j$ is the path from landmark L_j to $\hat{e}_j$. The tree structure $\mathcal{T}_m = (\mathcal{L}_m, \mathcal{E}_m)$ is the *kinematic roadmap* of the manipulator $\mathcal{A}$. A schematic representation of $\mathcal{T}_m$ is shown in Figure 1. Note that the cardinality of $\mathcal{L}_m, \mathcal{E}_m, \mathcal{PL}_m$ and $\mathcal{PE}_m$ is m. At times, we may omit the sub-index m and simply write $\mathcal{T}$. $EXPLORE$ iteratively builds $\mathcal{T}$ by placing one new landmark at each iteration.

The set of landmarks is initialized to $\mathcal{L}_1 = \{L_1 = \hat{q}_0\}$. $EXPLORE$ generates a random trajectory $\hat{\gamma}_1$ and the end of this trajectory is initialized to the first embryo, i.e., $\mathcal{E}_1 = \{\hat{e}_1 = \hat{\gamma}_1(1)\}$. The edge set of the tree is initialized to null set, i.e., $\mathcal{PL}_1 = \phi$, and $\mathcal{PE}_1 = \{\hat{\gamma}_1\}$. At step m, the embryo, say $\hat{e}_k \in \mathcal{E}_m$ that is *farthest* from $\mathcal{L}_m$, (i.e., $d(\hat{e}_k, \mathcal{L}_m) \geq d(\hat{e}_j, \mathcal{L}_m), \forall \hat{e}_j \in \mathcal{E}$), becomes the $(m+1)^{th}$ landmark, i.e., $L_{m+1} = \hat{e}_k$. The corresponding path, $\hat{\gamma}_k$ is added to the edge set, i.e., $\hat{\varphi}_{m+1} = \hat{\gamma}_k$. Two new embryos are generated – one embryo to replace the embryo that just became a landmark and the the other from this newly generated landmark. Each of these is obtained, as before, by executing a random path. The first path, $\hat{\gamma}_k'$ is generated from L_k and the end of the path becomes an embryo, i.e., $\hat{e}_k' = \hat{\gamma}_k'(1)$. The second path, $\hat{\gamma}_{m+1}$ is generated from L_{m+1} and the end of this path becomes the corresponding embryo, i.e., $\hat{e}_{m+1} = \hat{\gamma}_{m+1}(1)$

$EXPLORE(m)$ takes the roadmap $\mathcal{T}_m$ as its input and returns an updated roadmap $\mathcal{T}_{m+1}$ and is stated in pseudocode form as follows :

```
EXPLORE(m)
begin
    max_dist = 0
    for i = 1 to m
        min_dist = ∞
        for j = 1 to m
            distance = ||ê_i − L_j||
            if (distance < min_dist)
                min_dist = distance
        if (min_dist > max_dist)
            k = i
    L_{m+1} = ê_k
    φ̂_{m+1} = γ̂_k
    γ̂_{m+1} = random_path_from(L_{m+1})
```

$\hat{\gamma}_k = random\_path\_from(L_k)$
$\hat{e}_{m+1} = \hat{\gamma}_{m+1}(1)$
$\hat{e}_k = \hat{\gamma}_k(1)$
end

where the $random\_path\_from(\hat{q})$ generates a random path beginning at configuration $\hat{q}$.

4.1 Generating Random Paths

4.1.1 Manhattan paths and Bouncing Technique

As we have shown in the previous section, each of the landmarks generated by the *EXPLORE* algorithm is obtained by generating a random path. In order to do this we use a special class of paths, the *Manhattan paths*, which consist of moving one robot link at a time. Formally, a Manhattan path $\hat{\gamma}$ is completely defined by

$$\hat{\gamma} = (\Delta_1, \Delta_2, ..., \Delta_n)$$

Note that the semantic of this path is "move link 1 a distance Δ_1" followed by "move link 2 a distance Δ_2", and so on. Furthermore, the product of ℓ single Manhattan paths is a Manhattan path of order ℓ.

It is clear that all valid Manhattan paths of order k can be represented by a vector $\hat{x} \in I\!R^{n*k}$ but not all $\hat{x} \in I\!R^{n*k}$ represent a valid Manhattan Path, since $\Delta_i \in \hat{\gamma}^k$ could easily violate a kinematic constraint, either due to an obstacle or due to joint limit. The *bouncing technique* allows us to map an arbitrary Manhattan path into an *admissible* Manhattan path. The basic idea is to *bounce off* the obstacle and has been used in [BA+93, HFT94] (see Figure 5). Suppose a certain Δ_i leads to collision. As mentioned before, at the given configuration, a collision-free interval $[\Delta_i^{min}, \Delta_i^{max}]$ can be easily obtained (as in [LP87]) such that every joint value within this interval is collision-free (see Figure 3). We can then map the original interval Δ_i *modulo* the collision-free interval to within the collision-free interval $[\Delta_i^{min}, \Delta_i^{max}]$. Physically speaking, it is as if the the robot link repeatedly bounces off the obstacles until the entire Δ_i is travelled. Obviously, this bouncing move is not executed by the robot, it is just a mapping from $\Delta_i \in I\!R$ to $[\Delta_i^{min}, \Delta_i^{max}]$. Note that the range of link $(i+1)$ depends on the moves of link $1 \ldots i$, i.e., Δ_{i+1}^{min} and Δ_{i+1}^{max} are functions of Δ_j for $j = 1, 2, \ldots, i$. *We can now generate a random Manhattan path (by randomly generating each Δ_i, which is equivalent to generating a random vector $\hat{x} \in I\!R^{n*k}$) and then transform it into an admissible Manahttan path using the bouncing technique.*

5 Algorithm *INVKIN*

Using *SEARCH* and *EXPLORE*, our inverse kinematics algorithm *INVKIN* is obtained as follows:

$INVKIN(\hat{q}_0, \hat{q}_{\mathcal{G}}, n\_iterations)$
begin `[initialization of` $\mathcal{T}$`]`
  $m = 1$
  $L_1 = \hat{q}_0$
  $\hat{\gamma}_1 = random\_path\_from(L_1)$
  $\hat{e}_1 = \hat{\gamma}(1)$
  $path\_found = false$
  repeat `[place landmarks until solution]`
      `or` m `reaches` $n\_iterations$`]`
    if $(SEARCH(L_m, \hat{x}_g) == 0)$
      $path\_found = true$
    else
      $EXPLORE(m)$ `[increment` $\mathcal{T}$`:add landmark]`
      $m = m + 1$
  until($m > n\_iterations$ or $path\_found$)
  if $(path\_found)$
    return$(g^{loc}(\hat{q}))$
  else
    return$(fail)$
end

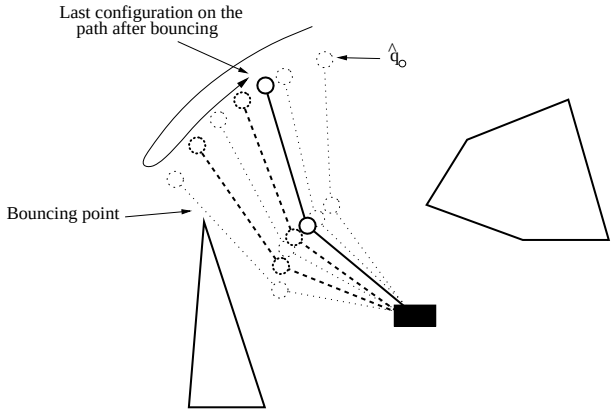


Figure 5: A physical illustration of bouncing technique.

6 Experimental Results

The above algorithm, *INVKIN* has been implemented (on an SGI Indigo II) for a 7-*dof* manipulator arm in 3-dimensional environments. All joints are revolute and each joint has a joint limit (e.g., -90 deg to 90 deg for joint 1, and similar limits for other joints). Polyhedral CAD models were used for both the robot and the environment. The collision detection scheme is based on the algorithm in [LP87] that efficiently computes a collision-free interval for a given robot link as it rotates, keeping all other degrees of freedom fixed. These collision-free intervals are then used by the bouncing technique to convert an arbitrary Manhattan path into a collision-free Manhattan path. Several standard techniques – simplified and hierarchical representations for manipulator links and obstacles, etc. – were employed for efficient collision detection. The robot links, the fixed obstacles, and the payload are each represented by a parallelepiped.

Figure 6 illustrates a difficult example with a few fixed obstacles present in the environment. The initial configuration of the arm, and the desired end-effector location is shown in Figure 6a. Note that the desired location of the end-effector is under the table and therefore, quite constrained. Small moves of the robot must be executed to achieve this (if it is indeed possible). 58 landmarks were needed to solve this example and total execution time was 30 seconds.

In order to extensively test our algorithm, we first generated several random and collision-free configurations of the robot and saved the corresponding configurations of the end-effector. This was done so that we knew that indeed there is a manipulator configuration corresponding to the desired end-effector location. These end-effector configurations were then used as goal configurations to obtain various test statistics such as percentage success, average number of land-

marks placed, and average run time. For environments free of obstacles, the average number of landmarks placed was 61 (ranging between a minimum of 1 and maximum of 372) and average run time was 16.9 seconds (minimum 0.3 seconds, maximum 80 seconds). Planner *always* found a solution. For environments with obstacles, the average number of landmarks placed was 62 (minimum 1 and maximum 353) and average run-time was 34.5 seconds (minimum 0.3 seconds and maximum 199 seconds).

7 Discussion

There are several interesting issues to be explored further. A main one is to exploit the topology of the configuration space while creating this kinematic roadmap. It is well known that the configuration space can be sub-divided into *c-bundles* separated by co-regular surfaces. Another important issue to be explored is the resolution completeness of our approach.

References

[Ahu94] J.M. Ahuactzin. *Le Fil d'Ariane. Une méthode de planification générale. Application à la planification automatique des trajectoires.* PhD thesis, Institut National Polytechnique de Grenoble, France, 1994.

[BA+93] Pierre Bessiere, Juan Manuel Ahuactzin, et al. The "ariadne's clew" algorithm: Global planning with local methods. In *IEEE/RSJ conference on Intelligent Robots and Systems*, 1993.

[Bur89] Joel Burdick. On the inverse kinematics of redundant manipulators: Characterization of the self-motion manifolds. In *ICRA*, pages 264–269, 1989.

[CH92] P. C. Chen and Y.K. Hwang. Sandros: A motion planner with performance proportional to task difficulty. In *IEEE ICRA*, 1992.

[Cra91] John J. Craig. *Introduction to Robotics.* Addison Wesley, 1991.

[Gup90] Kamal Kant Gupta. Fast collision avoidance for manipulator arms: A sequential search strategy. *IEEE Transactions on Robotics and Automation*, 6(5):522–532, july 1990.

[GZ95] Kamal Kant Gupta and Xinyu Zhu. Practical global motion planning for many degrees of freedom: A novel approach within sequential framework. *Journal of Robotic Systems*, 2(12):105–118, 1995.

[HFT94] Thomas Horsch, F.Scwarz, and H. Tolle. Motion planning for many degrees of freedom – random reflections at c-space obstacles. In *ICRA*, pages 3318–3323, 1994.

[KL94] Lydia Kavraki and Jean-Claude Latombe. Randomized preprocessing of configuration space for path planning: Articulated robots. In *IROS*, 1994.

[KR92] K. Kazerounian and J. Rastegar. Object norms: A class of coordinate and metric independent norms for displacements. In G. Kinzel et al., editors, *Flexible Mechanisms, Dynamics and Analysis.* ASME DE-Vol 47, 1992.

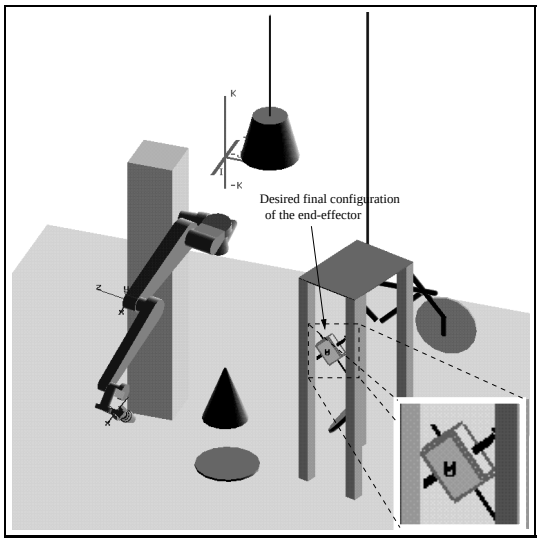
[Lat91] Jean-Claude Latombe. *Robot Motion Planning.* Kluwer Academic, 1991.

[LL94] Carlos Lück and Sukhan Lee. Global path planning of redundant manipulators based on self-motion topology. In *ICRA*, pages 372–377, 1994.

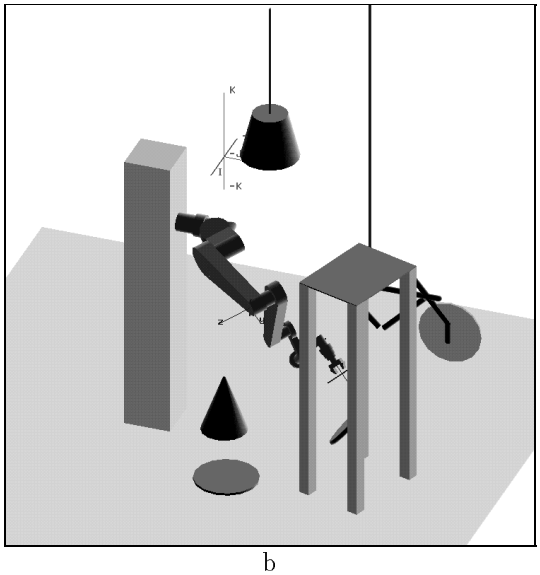
[LP87] Tomas Lozano-Perez. A simple motion planning algorithm for general robot manipulators. *IEEE Transactions on Robotics and Automation*, 3(3), 1987.

[Par95] F.C. Park. Distance metrics on the rigid-body motions with applications to mechanism design. *ASME Journal of Mechanism Design*, 117:48–54, 1995.

[Sic90] Bruno Siciliano. Kinematic control of redundant manipulators: A tutorial. *Journal of Intelligent and Robotic Systems*, 3:201–212, 1990.



a



b

Figure 6: Inverse kinematics in the presence of fixed obstacles. The initial robot configuration and the desired end-effector configuration are shown in (a). A zoomed-in view of the desired end-effector configuration is shown in the bottom right sub-square in (a). The solution determined by our algorithm is shown in (b).